## Remarks

This REPLY is in response to the Office Action mailed September 20, 2007. A Petition for Extension of Time is included herewith, together with the appropriate fee. No fee is due for the addition of new claims.

### I. Summary of Examiner's Rejections

Prior to the Office Action mailed September 20, 2007, Claims 1, 4, 6-8, 10, 11, 14, 16-18, 20, 21, 24, 26-28 and 30-42 were pending in the Application. In the Office Action, the Specification and Claims 34, 37 and 40 were objected to for various informalities, including the use of trademarks. Claims 31-33 were rejected under 35 U.S.C. 112, second paragraph as being indefinite for failing to particularly point out and distinctly claim the subject matter which Applicant regards as the invention. Claims 1, 4, 6-8, 10, 11, 14, 16-18, 20, 21, 24, 26-28 and 30-42 were rejected under 35 U.S.C. 103(a) as being unpatentable over Berg et al. (U.S. Publication No. 2004/0088681, hereafter Berg) in view of Rich et al. (U.S. Publication No. 2002/0178439, hereinafter Rich), and further in view of McIntyre (U.S. Patent No. 6,178,546).

### II. Summary of Applicant's Amendments

The present Response amends the Specification, and Claims 1, 11, 21, 31-34, 37 and 40; leaving for the Examiner's present consideration Claims 1, 4, 6-8, 10, 11, 14, 16-18, 20, 21, 24, 26-28 and 30-42.

### III. Objections to the Specification

In the Office Action mailed September 20, 2007, the Specification and Claims 34, 37 and 40 were objected to for various informalities, including the use of trademarks. Accordingly, the Specification and Claims 34, 37 and 40 have been amended as shown above. Applicant respectfully submits that the proposed amendments correct informalities in the Specification and that no new matter is being added. Reconsideration thereof is respectfully requested.

### IV. Claim Rejection under 35 U.S.C. §112

In the Office Action mailed September 20, 2007, Claims 31-33 were rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to point out and distinctly claim the subject matter that Applicant regards as the invention. Accordingly, Claims 31-33 have been amended as shown above. Applicant respectfully submits that Claims 31-33, as amended, conform to the requirements of 35 U.S.C. 112, and reconsideration thereof is respectfully requested.

## V. Claim Rejections under 35 U.S.C. §103(a)

In the Office Action mailed September 20, 2007, Claims 1, 4, 6-8, 10, 11, 14, 16, 18, 20, 21, 24, 26-28 and 30-42 were rejected under 35 U.S.C. 103(a) as being unpatentable over Berg (U.S. Publication No. 2004/0088681) in view of Rich (U.S. Publication No. 2002/0178439), and further in view of McIntyre (U.S. Patent No. 6,178,546).

### Claim 1

Applicant thanks the Examiner for the detailed comments provided in the Office Action with respect to Claim 1, which has been amended by the current Reply to more clearly define the embodiment therein. As amended, Claim 1 defines:

> *1. (Currently Amended) A system for organization of software application files during development and subsequent deployment of the software application to a server, comprising:*
>
> *a split directory structure stored on a computer medium that stores files for a software application, wherein, for each software application, the split directory structure includes both a source folder that stores editable source files as part of the software application, and a corresponding output folder that stores system-generated compiled versions of the source files as part of the software application, together with a build file that indicates the output folder as being an output of its corresponding source folder, and wherein the split directory is accessed as a virtual file that provides an abstraction over the two folders therein, so that the two folders appear to a server as a single application;*
>
> *a server upon which the software application will be deployed; and*
>
> *a deployment tool that allows a user to specify the output folder during deployment of the software application, wherein during the deployment of an application class or resource the server interprets the software application as a union of both the source folder*

*and the output folder, recognizes the split directory structure, parses the build file to determine the corresponding source folder, and deploys the application by making requests to the virtual file which checks first the source folder for the class or resource, and then the corresponding output folder for that class or resource, before deploying the software application files to the server.*

Claim 1, as currently amended, defines a split directory structure that stores files for a software application, wherein, for each software application, the split directory structure includes both a source folder that stores editable source files as part of the software application, and a corresponding output folder that stores system-generated compiled versions of the source files as part of the software application, together with a build file that indicates the output folder as being an output of its corresponding source folder, and wherein the split directory is accessed as a virtual file that provides an abstraction over the two folders therein, so that the two folders appear to a server as a single application. During the deployment of an application class or resource the server interprets the software application as a union of both the source folder and the output folder, recognizes the split directory structure, parses the build file to determine the corresponding source folder, and deploys the application by making requests to the virtual file which checks first the source folder for the class or resource, and then the corresponding output folder for that class or resource, before deploying the software application files to the server.

Berg discloses a system for dynamically mapping archive files in an enterprise application. Mapping information that describes the mapping of referenced projects to their container project is included in the container project, using a "module mapping" file. (Abstract). When the developer starts the application server within the test environment, the development tool reads the module map file at that time and computes the absolute addresses of the nested archive files, which are represented by referenced projects, relative to the environment (e.g., the machine) in which the application server is running. (Paragraph [0020]). Since the computed absolute paths are stored in a loose configuration file that is thus outside of the container project, the container project itself is not modified. Therefore, the application can be tested/run on multiple machines without the need to modify the container project. (Paragraph [0021]).

Rich discloses a method and system for providing a programming interface for loading and saving archives in enterprise applications. As disclosed therein, the two environments, assembly and "runtime" (the running application server), deal with two different physical file structures: assembly, with JARs; and runtime, with a mixture of expanded JARs in a directory tree structure and JAR files themselves. (Paragraph [0014]). The user (running program) requests the loading of an archive having a given file path/name. A determination is made if the requested file is an archive file type (e.g., .jar, .zip, .war, etc.) or a directory tree file. A loading strategy for loading and displaying the file requested by the user ("open archive" command) is created based on the status of the file requested as having an archive structure or a directory tree. (Paragraphs [0025-0028]).

McIntyre discloses a method of making software product deliverables includes reading a description file having descriptions of items to be built, packaged, and/or installed, identifying the items, and then generating a build script to build the items described in the description file. An inventory of the items to be built and their respective locations in a build area is generated as well as a packaging list of the items to be included in each package. (Abstract).

In the Office Action it was submitted that, in Berg, the container project could be considered to represent a split directory structure for an application, and that the container project (characterized in the Office Action as a split directory structure) includes a source folder that stores editable source files. It was further submitted that, in Berg, the utility JAR stores compiled files, and that this could be considered to represent an output folder, because its contents are expanded into a directory; and that the utility JAR is included in the container project and could be considered to represent a corresponding output folder, because it corresponds to the same container projects as the source folder. It was further submitted that Rich discloses a virtual archive that provides an abstraction over a directory structure; and that McIntyre discloses first checking one build area for software application files, and, if the files are not found in the build area, then checking another build area.

However, Applicant respectfully submits that, while Berg appears to disclose that original files are stored in an IDE, there does not appear to be any description in Berg as to how files that are generated or compiled are then accessed together with their corresponding source files in a split directory structure. Instead, Berg appears to disclose that a higher-level or container *project* can be mapped to other, referenced *projects*. Similarly, Applicant respectfully submits that, in Rich, a determination appears to be made depending on whether the requested file is an *archive file type* or a *directory tree file*. Additionally, Applicant respectfully submits that the technique described in McIntyre appears to be one of arranging and using a variety of *build areas*, for use in packaging a software application, rather than, say, a combination of editable source files and their corresponding compiled files.

Claim 1 has been amended to more clearly define the embodiment therein as comprising a split directory structure that stores files for a software application, wherein, for each software application, the split directory structure includes both a source folder that stores editable source files as part of the software application, and *a corresponding output folder that stores system-generated compiled versions of the source files*. The output folder stores a *build file* that indicates the output folder as being an output of its corresponding source folder. During the deployment of an application class or resource the server parses the build file to determine the corresponding source folder, and deploys the application by making requests to the virtual file which checks first the source folder for the class or resource, and then the corresponding output folder *for that class or resource*. Applicant respectfully submits that these features are neither anticipated nor rendered obvious by the cited references.

Furthermore, Applicant respectfully submits that the proposed combination of reference appears to be directed to a different problem, and different goals, from the embodiment defined by Claim 1. In particular, each of Berg, Rich, and McIntyre appear to be directed to a means of sharing application projects, file formats, or build areas, to provide an ability to share application development across different machines and/or different teams of developers (as described for example in Berg: Method allows files to be easily tested without modification, enables full access to members of a development team without requiring that files be modified. Paragraph [0039]; Rich: Invention relates to [ ] interfaces for providing simplified access to enterprise applications on which files are stored in multiple formats, such as archive format and directory-tree format. Paragraph [0001]; and McIntyre: Developers can update their own libraries and programs and still reference other developer's files in the public build area. Column 2, lines 60-63).

In contrast, the embodiment of Claim 1 provides a means of supporting development of a software application, while presenting a clean separation between human-readable source files and generated java class files, in that the software application is interpreted as a union of both the source folder and the output folder. As a result, there is no need for copying of files; instead the server can read source files directly from the split directory structure, without having to first copy them to a build directory. For example, Web files can be changed and redeployed in place within the source folder, without having to rebuild the entire software application. Applicant respectfully submits that since the cited references appear to address a different problem from that addressed by Claim 1, it would not have been obvious of one of ordinary skill in the art at the time the invention was made to combine the teachings of Berg, Rich, and McIntyre, in the manner described to anticipate or to render obvious the embodiment defined by Claim 1.

In view of the above comments, Applicant respectfully submits that Claim 1, as amended, is neither anticipated by, nor obvious in view of the cited references, and reconsideration thereof is respectfully requested.

**Claims 11 and 21**

The comments provided above with respect to Claim 1 are hereby incorporated by reference. Claims 11 and 21 have been similarly amended by the current Response to more clearly define the embodiments therein. For similar reasons as provided above with respect to Claim 1, Applicant respectfully submits that Claims 11 and 21, as amended, are likewise neither anticipated by, nor obvious in view of the cited references, and reconsideration thereof is respectfully requested.

**Claims 4, 6-8, 10, 14, 16-18, 20, 24, 26-28 and 30-42**

Claims 4, 6-8, 10, 14, 16-18, 20, 24, 26-28 and 30-42 are not addressed separately but it is respectfully submitted that these claims are allowable as depending from an allowable independent claim, and further in view of the comments provided above. Applicant respectfully submits that Claims 4, 6-8, 10, 14, 16-18, 20, 24, 26-28 and 30-42 are similarly neither anticipated by, nor obvious in view of the cited references, and reconsideration thereof is respectfully requested.

## VI. Conclusion

In view of the above amendments and remarks, it is respectfully submitted that all of the claims now pending in the subject patent application should be allowable, and reconsideration thereof is respectfully requested. The Examiner is respectfully requested to telephone the undersigned if he can assist in any way in expediting issuance of a patent.

Enclosed is a PETITION FOR EXTENSION OF TIME UNDER 37 C.F.R. §1.136 for extending the time to respond up to and including January 22, 2008.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 06-1325 for any matter in connection with this reply, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: January 22, 2008

By: [illegible]
Karl Kenna
Reg. No. 45,445

Customer No.: 23910
FLIESLER MEYER LLP
650 California Street, 14th Floor
San Francisco, California 94108
Telephone: (415) 362-3800